Project Report: License Plate Detection Recognition Using WPOD-NET and OCR

Prathmesh Patel

Electrical and Computer Engineering at North Carolina State University, Raleigh, USA

Code available at: https://github.com/patelp1003/LPDR

***Abstract*—There are many applications that require cars to be identified and detected. Parking, toll collection, and security surveillance are some areas that actively use video cameras to locate and track cars. However, many of the devices used for these applications are still not equipped with the latest machine learning technology. Additionally, camera systems that automatically detect license plates are not widespread and adopted everywhere. This project attempts to create a computer vision system which can be used for License Plate Detection and Recognition (LPDR), and tests its feasibility to work on IoT edge devices. The purpose of this report is to highlight the system structure and algorithm that was developed, the findings and learnings, and the results.**

Keywords: LPDR, OCR, license plate capture, IoT

I. Introduction

IN The United States of America, cars are the primary mode of transportation. With such a large number of vehicles on the roads and highways, the applications and use cases of video tolling are enormous.

However, the implementation of this technology is not as large as the need. Many places, such as parking lots, still use traditional ways to identify and keep track of cars, which can be easily enhanced with computer vision.

With the increase in progress and participation in the Machine Learning community, and the greater adoption of Internet of Things (IoT) devices, this project will aim at creating a License Plate Detection Recognition (LPDR) algorithm which can be implemented on an IoT device, such as a Raspberry Pi [7].

The project will be divided into several milestones, as shown below.

- Achieving license plate identification and extraction
- Detecting and recognizing the text
- Deployment and test

For the purposes of initial testing, the model was deployed and successfully tested on a computer with a webcam.

II. Structure and Algorithm

The basic steps of the algorithm are described in the following diagram.

To detect license plate, a pretrained convolution neural network (CNN), WPOD-NET [1], was employed. It is made up of 21 convolutional layers and is based on the YOLO architecture. The developers tweaked the end layers and introduced a linear layer with the original softmax to achieve a balance between accuracy and efficiency, as shown in Fig. 1 [2].

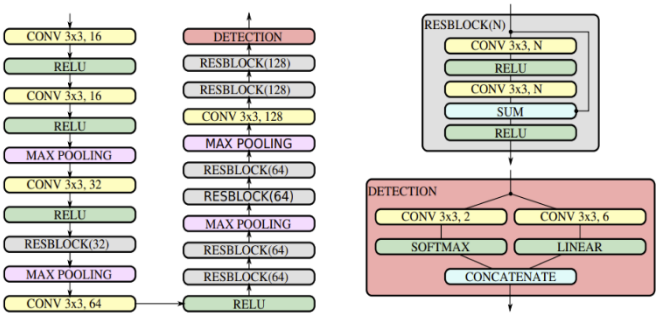

Fig. 1. Detailed WPOD-NET Architecture

Due to the specifications of the network, any image that needs to be passed must be rescaled to fit the configuration [1]. As an image is passed to the network, it detects and outputs the coordinates of the region of interest, which has the highest probability of the location of the plate. An example is shown in Fig 2.

Fig. 2. Output of the WPOD-NET [4]

Fig. 3. Output of text recognition by PP-OCR model [5]

Fig. 4. Standard license plate from Vietnam (left) vs standard license plate from the USA (right)

This network was downloaded and run using Keras [9] and Tensorflow [8], and the preprocessing of the image, as well as visualization of the result, was done using OpenCV [10]. The tutorial provided in reference [3] was used to implement the model and process the images.

After isolating the license plate image from the overall picture of the car, comes the task to recognize the text in the image. Given the large amount of work done in the area of developing Optical Character Recognition (OCR) models, using an efficient OCR model, such as PP-OCR [5], would be a suitable option.

PP-OCR is an extremely lightweight OCR model specially designed for small-scaled computing and IoT devices. Since the eventual goal was to deploy the program on an edge device, this model was chosen over some of the other verified and reviewed models. The model consists of three parts: text detection, detection box rectification, text recognition. Text detection is achieved through differentiable binarization which locates the area of the text in the image, detection box rectification rearranges the box into a horizontal rectangle to better fit the text recognition model, and a Convolutional Recurrent Neural Network (CRNN) is used for text recognition [5]. An example of the output is shown in Fig. 3, which also showcases the model being able to recognize text in different angles and orientations. This feature will be useful when working with moving objects.

Applying this model to our application, the image from the WPOD model is passed on to the OCR model and the output text is recorded. Since American license plates also contain background text, a function is implemented to find the text with the largest area, which is used as the prediction of the license plate. The prediction from the model will be displayed as red text on the plate image that was captured. The guide supplied in [6] was used to develop the code for the text recognition.

III. Findings and Learnings

This project has provided various learnings and inspiration to further explore the computer vision field. Since many existing online resources were geared towards detecting plates with single text blocks, finding and constructing an algorithm to detect American license plates was a motivating challenge to explore. As it can be seen in Fig. 4, many license plates around the world only comprised of the number itself, with no text or imagery in the background. Therefore, using the WPOD-Net model, which could detect American plates, was a promising option to research and pursue.

Before using PP-OCR, some attempts to read text included using PyTesseract and OpenCV. PyTesseract is one of the first optical character recognition libraries which has implementation methods in Python. It takes an image as an input, and identifies and supplies the probable text that can be seen in the image. This method was employed, but due an excessive amount of text and symbols in the background, identifying the desired license plate number was not achieved.

Another option, which was described in [3], was to process the image by applying the gaussian blur function, turning the image grey, and finding contours with specific ratios to indicate where the text is. This proved to be an effective method for some license plates, as seen in the Fig 5.

Fig. 5. Using the blur and contour filtering technic with OpenCV

However, this would require tweaking of the filtering parameters based on individual images, which would not be reliable in the long run.

In the end, the PP-OCR model proved to be the most effective in recognizing the text. As seen in Fig. 6, the model can identify many of the texts that are present on the license plate. Even with a lower resolution and some of the text being unclear, the model is still able to identify the number correctly, as seen in Fig. 7. Therefore, for recognizing the text of the plate, this option proved to be the most efficient and reliable out of all other choices.

Fig. 6. Multiple texts being identified by PP-OCR

Fig. 7. Text being identified by PP-OCR in a blurry image

The last part of the project was to package the program, and deploy the working model on a Raspberry Pi. Due to limitations of TensorFlow on the ARMv7 [1] architecture for the latest version of Python, the initial testing of the algorithm was done on a Conda environment on a computer, with the aim to deploy it on the Pi as the next step.

Coming to the experience gained, a personal learning that would assist beyond the scope of this project, is realizing how to work with established models. It may not be always feasible to train accurate models with limited resources. Hence, learning how to use models already published is of great value. It can help later with other projects, and will also pave the way for learning how to create custom models in the future.

IV. Final Results

The final results of the project are compiled and shown below, for several images and test cases. It comprises the output of plate recognition, as well the recognition of the text. As it can be seen in Fig. 8, there were occasional inaccuracies due to the presence of character symbols, but the overall performance of the model was satisfactory and worked as expected. Images taken from different angles and clarity were tested, with the model being able to successfully predict the output. Additionally, plates from different states were also tested, with each having variation in spaces and characters, and most of the results seemed to align with the expected license plate number as well.

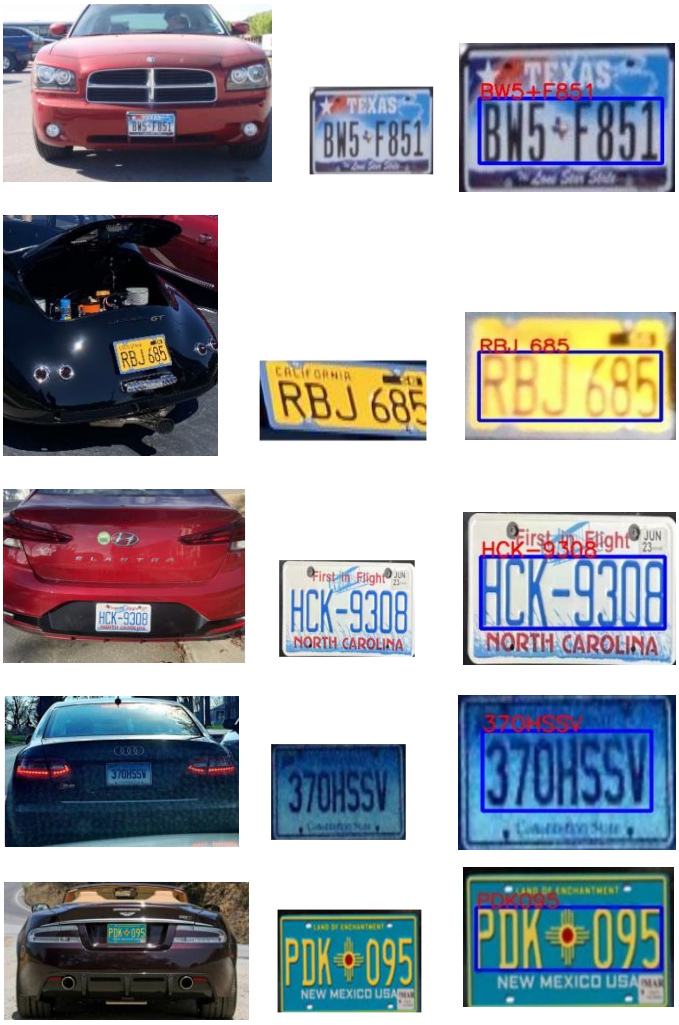

Fig. 8. Output of the model tested on different images

REFERENCES

[1] P. N. Huu, C. V. Quoc, T. N. Vu, H. N. Trong, Q. T. Minh and T. N. Dac, "Proposing algorithm for detecting car number plate using SVM and WPOD-NET models," 2020 International Conference on Advanced

[2] Computing and Applications (ACOMP), 2020, pp. 29-33, doi: 10.1109/ACOMP50827.2020.00012.

[3] S. M. Silva και C. R. Jung, 'License Plate Detection and Recognition in Unconstrained Scenarios', στο *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[4] Q. Nguyen, "Detect and Recognize Vehicle's License Plate with Machine Learning and Python — Part 2: Plate…," *Medium*, May 17, 2020. https://medium.com/@quangnhatnguyenle/detect-and-recognize-vehicles-license-plate-with-machine-learning-and-python-part-2-plate-de644de9849f (accessed Nov. 11, 2022).

[5] Y. DU *ET AL.*, 'PP-OCR: A PRACTICAL ULTRA LIGHTWEIGHT OCR SYSTEM'. ARXIV, 2020.

[6] ANDYJPADDLE, 'PADDLEOCR', GITHUB REPOSITORY. GITHUB, 2022.

[7] RASPBERRY PI, RASPBERRY PI. [ONLINE]. AVAILABLE: HTTPS://WWW.RASPBERRYPI.COM/. [ACCESSED: 04-FEB-2023].

[8] "TENSORFLOW," TENSORFLOW. [ONLINE]. AVAILABLE: HTTPS://WWW.TENSORFLOW.ORG/. [ACCESSED: 04-FEB-2023].

[9] K. TEAM, "SIMPLE. FLEXIBLE. POWERFUL.," KERAS. [ONLINE]. AVAILABLE: HTTPS://KERAS.IO/. [ACCESSED: 04-FEB-2023].

[10] OPENCV, 31-JAN-2023. [ONLINE]. AVAILABLE: HTTPS://OPENCV.ORG/. [ACCESSED: 04-FEB-2023].

[11] HTTPS://DEVELOPER.ARM.COM/DOCUMENTATION/DDI0406/LATEST

[12] "CONDA ENVIRONMENTS," CONDA ENVIRONMENTS - CONDA 23.1.0.POST22+31FE66BED DOCUMENTATION. [ONLINE]. AVAILABLE: HTTPS://DOCS.CONDA.IO/PROJECTS/CONDA/EN/LATEST/USER-GUIDE/CONCEPTS/ENVIRONMENTS.HTML#:~:TEXT=A%20CONDA%20ENVIRONMENT%20IS%20A,NUMPY%201.6%20FOR%20LEGACY%20TESTING. [ACCESSED: 04-FEB-2023].

[13] PYTHON.ORG. [ONLINE]. AVAILABLE: HTTPS://WWW.PYTHON.ORG/. [ACCESSED: 04-FEB-2023].